IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 October 2023, accepted 22 December 2023, date of publication 5 January 2024,
date of current version 11 January 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3350381*

RESEARCH ARTICLE

# Video-Based Vehicle Speed Estimation Using Speed Measurement Metrics

**KEATTISAK SANGSUWAN AND MONGKOL EKPANYAPONG**
School of Engineering and Technology, Asian Institute of Technology, Khlong Nueng 12120, Thailand
Corresponding author: Keattisak Sangsuwan (st117815@ait.asia)

**ABSTRACT** Camera system is widely used as a road traffic monitoring nowadays but if the system is used as a speed camera, an additional speed sensor is required. In this work, we demonstrate a novel method to estimate speed of vehicle in the traffic video without using the additional sensor. We implement two speed measurement models which are measuring traveling distance of the vehicle in a given unit of time and measuring traveling time in a given unit of distance. To get parameters of the models, we define four virtual intrusion lines on road in the camera view. Then, YOLOv3, DeepSORT, GoodFeatureToTrack, and Pyramidal Lucas-Kanade optical flow algorithm are implemented together to detect and track the target vehicle while moving in the camera view. From the tracking data, pixel displacement between two consecutive frames (before and after the vehicle crossing the line) is measured as Crossing distance. The number of frames that the vehicle uses while moving from the first line to the other lines is measured as Traveling time. These two parameters at each intrusion line are used as speed measurement metrics. Solution of the metrics are solved by using tracking data of 20 vehicles at 9 different ground truth speeds measured by a laser speed gun. Then, the metrics are used to estimate speed of 813 vehicles. Our best accuracy is with MAE of 3.38 and RMSE of 4.69 km/h when comparing to their ground truth speed. The same dataset are tested on a Multilayer Perceptron Neural Network model. It can reach accuracy with MAE of 3.07 km/h (RMSE 3.98 km/h).

**INDEX TERMS** Vehicle speed estimation, DeepSORT, YOLOv3, GoodFeatureToTrack, Lucas-Kanade, optical flow, vehicle tracking.

## I. INTRODUCTION

Vehicle speed measurement can be categorized into two groups including intrusive and non-intrusive technologies [1], [2]. For intrusive technologies, the speed sensors are embedded under the road surface. These sensors, for examples, are Inductive loop detectors, Magnetic detector, Piezo-electric, Weight-In-Motion. Installation and maintain of these sensors are costly and difficult. For non-intrusive technologies, the speed sensors can be divided into overhead mounted and side-fired sensors. The sensors are, for example, infrared speed sensor, microwave radar speed sensor, and laser speed sensor. These advanced sensors are costly but easy to install, access, and maintain [1], [2], [3]. However, both intrusive and non-intrusive technologies require a camera for applying as an automatic road speed limit violation system (called speed camera system) because the camera has to be used for taking a photo of the speed violation vehicle. On the other hand, many cameras (without speed sensor) are widely installed on highways, Toll ways, urban road, or intersections in order to be used for traffic monitoring or traffic management purpose [4]. Therefore, it would be a great advantage if these cameras can be integrated with speed measurement function without additional speed sensor.

Recent advance computer vision technologies introduce numerous studies and research in vision-based vehicle speed measurement. The main effort is trying to adapt the traffic monitoring camera to be used as a speed camera without

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng Yan.

using additional speed sensor. With many proposed solving techniques over this problem, the solution can be broken down into three main groups [5] including, camera settings and calibration techniques, vehicle detection and tracking technique, and the technique of ground truth speed generation for accuracy evaluation. Camera settings and calibration techniques require complicated work to get parameters for transferring between 2D camera image plane ($x$, $y$) and 3D real-world coordinate system ($X_w$, $Y_w$, $Z_w$) [6], [7], [8], [9]. Vehicle detection and tracking technique can be done by various approaches including background/foreground subtraction [10], [11], [12], feature-based detection [6], [7], [13], [14], machine learning-based detection [15], [16], [17], [18], [19], [20]. Ground truth speed generation is the work to generate data for evaluating the proposed speed measurement model [16]. The data can be generated by the existing vehicle speed measurement system both intrusive or non-intrusive technologies. In summary, integrating speed measurement function into the camera is a complicated task and it still be a challenge problem to be solved for a great potential advantage.

In this work, we focus on speed estimation of moving vehicle in the video frames received from a fixed Field of View (FoV) traffic camera system. The goal is to provide a method which is easy to integrate speed measurement function into the system without using additional speed sensor.

Our proposed process starts by applying a Deeplearning-based computer vision technology, YOLOv3 [21] and Simple Online Real-time Tracking with Deep association metric algorithm (DeepSORT) [22], to detect and track the target vehicle while GoodFeatureToTrack [23] and pyramidal Lucas-Kanade optical flow [24] algorithms are supported to increase tracking precision by tracking the vehicle at wheel area. These algorithms together with four virtual intrusion lines on the videos frame are used to generate speed measurement metrics and then the metrics is used to estimate speed of the other vehicle in the same FoV camera. The main contributions of this work are:

1) We show that YOLOv3 with DeepSORT and feature tracking at wheel area improves vehicle tracking accuracy than using only YOLOv3 detection box.

2) We introduce two speed measurement metrics with simple machine learning method techniques to obtain higher speed estimation accuracy.

3) We show that the metrics for speed measurement reduce error from fault detection.

4) We propose an input metric for vehicle speed estimation by using a simple machine learning model.

5) We analyze impact and errors arising from two speed measurement metrics and arising from changing of video frame rate.

The remainder of this paper is organized as follows. Section II presents some related works on vehicle speed estimation with different computation techniques. Section III presents our proposed method for vehicle speed estimation. Section IV presents the experimental results. Finally, Section V presents the conclusion of this work.

## II. RELATED WORKS

Vision-based vehicle speed measurement is a work to analyze a series of consecutive traffic video frames. The vehicle must be detected first and keep tracking in the following frame to get speed in pixel per seconds. Then, the speed on the image plane is converted to real-world speed (km/h) by using pixel-to-meter relationship (called camera perspective or geometric transformation). In this section, we discuss the existing vehicle detection and tracking techniques.

### A. BACKGROUND-FOREGROUND BASED

Vehicle detection using background-foreground technique is a computer vision technology to detect vehicle in traffic video frames by using frame difference. The technique is the subtraction between reference frame and current frame. The reference frame with no vehicle is considered as a background image while another frame with a vehicle is considered as the current frame. Subtraction between these two frames results in a foreground image showing location of the detected vehicle (as a group of different pixels). Then, a series of the foreground image can be generated by subtracting the background with each traffic video frame. Later, the traveling distance of the detected vehicle between frames (or pixel displacement) can be calculated by subtracting ($x$, $y$) values among the foreground image. Where ($x$, $y$) is the horizontal and vertical pixel location which represent the geometric center of the detected vehicle (called centroid). Finally, the traveling distance in pixel unit compared to video frame rate in seconds can be converted to km/h by using the pixel-to-meter ratio.

In the study of Genyuan et al. [11], they used a simple machine learning algorithm such K-nearest Neighbor to model the background and then to detect vehicle on their traffic video. The centroid was used for vehicle tracking (Fig.1(a)). Finally, speed of the moving vehicle was estimated by using pixel-to-meter ratio referred to the road mark. The best accuracy was with the side view traffic video (comparing to top and intersection view). In [12], a near-top side view camera was used with background-foreground vehicle detection. By tracking the centroid, the speed estimation accuracy was 94% with in $+3/-2$ km/h of 1,870 vehicles in the speed range 0 to 79 km/h. However, detecting and tracking vehicle by using this technique is sensitive to background, light intensity, vehicle occlusion, and the other moving object in the frame.

### B. FEATURE BASED

A small point on the detected vehicle is hi-lighted for some studies. In [6], after the vehicle was detected by background-foreground technique, SURF [25] and FLANN [26] algorithms were implemented for vehicle feature detection and tracking. Although, they claimed that it was a robust method when facing light change conditions. However, they still

had a problem with vehicle occlusion. In [7], the moving vehicle was detected by background-foreground technique then some features on the vehicle license plate was selected and tracked by using GoodFeatureToTrack [23], Lucas-Kanade [24], and SIFT [27] algorithm (Fig.1(b)). Later, tracking distance in the pixel unit was transformed to meter by using camera perspective transformation. Finally, vehicle speed was estimated based on the video frame rate. The error value was reported at −0.5 km/h compared to the ground truth speed in range between 10 to 69 km/h detected by using inductive sensor. However, detecting and tracking vehicle on the license plate has some limitations, for example, the camera has to be installed in a near FoV, the high-resolution camera is required for good license plate detection, the height of license plate from ground level is sensitive to estimated speed, the license plate has to be in a good condition.

Another interesting feature detection was proposed in [28]. Instead of detecting feature of the vehicle, they detected a movement pattern vector of the vehicle in the traffic video. They systematically placed four intrusion lines on a single-lane road. Then, the passing vehicle was detected by using an optical flow algorithm. The passing frame number on each intrusion line was recorded as the vector. Finally, the probability distribution function model was applied to the vector for speed estimation. The maximum error 4% was reported compared to the ground truth speed measured by using GPS. However, a single-lane road is required and only a narrow speed range between 72 to 94 km/h was reported in the study.

### C. MACHINE LEARNING BASED

A recent computer vision technology on machine learning such as Deeplearning-based object detection algorithm [29] is also easily applied to vehicle detection [5]. The algorithm can be fed by a traffic video frame and, if there is only one vehicle on the frame, the output will be a rectangle box showing location of the detected vehicle on the image plane. The box is presented by four values ($x$, $y$, $width$, $height$). Where $x$ and $y$ represent respectively the top-left horizontal location and the top-left vertical location of the box. $width$ and $height$ are the size of the box in pixel unit.

Deeplearning-based object detection algorithms can be categorized into two types: Two-stage and One-stage. Two-stage type will find regions of the possible object at the first stage and then perform object classification on the second stage while these two stages are combined together for One-stage type. Among various proposed algorithms over these two types, One-stage type such RetinaNet [30] showed outstanding performance as illustrated in [21], but when testing with many traffic videos for vehicle detection, another One-stage type such YOLOv3 showed the lowest Floating Point Operation Per seconds (FLOPs) resulting in faster frame per seconds as illustrated in [31]. However, both types are applied for vehicle speed estimation with different techniques and some techniques can receive a good speed estimation accuracy as presented following.

Two-stage type such Faster-RNN [32] was applied in [16] to detect and track vehicle in traffic video. Tracking position of the 2D rectangle box given by the detector was not precise enough for speed estimation because the box position had too much variation (example in violet dot tracking line in Fig.1(c)). Then, the box was transformed to 3D bounding box and mapping to actual 3D box of the vehicle in database. By this way, size of the detected vehicle was estimated and tracked precisely with association of Kalman filter. Finally, the tracking distance in pixel unit and traveling time of the vehicle were used to estimate speed referring to pixel-to-meter ratio. Although, the study archived a good mean speed measurement error at 1.10 km/h but they used a complicated work processes and vehicle 3D model metadata were required for 2D to 3D bounding box transformation and mapping.

Another Two-stage type work presented in [33]. They used Kalman-based tracker called Simple Online and Realtime Tracking (SORT) [34] and DeepSORT [22] to track the bottom center of the box given by Mask-CRNN [35]. Finally, vehicle speed in pixel-per-seconds unit was transformed to real world unit in km/h by using camera perspective transformation. The error was reported at 15.35 km/h (9.54 mph).

One-stage type was in study of Bell et al. [17]. They used YOLOv2 [36] for vehicle detection. The bottom center of the box given by YOLOv2 was tracked by SORT (See the white van in Fig.1(c)). Then, the speed of was estimated by using camera perspective transformation. Their error was 2.25 km/h (0.25 m/s) but the samples were a few vehicle in urban traffic at low speed in range 1.6 to 15.8 km/h (0.452 to 4.393 m/s). Rangel et al. [18] used One-stage YOLOv3 algorithm to detect passing vehicle on side road and Kalman filter was used for tracking center of the box. Speed of the vehicle was estimated by linear regression models and machine learning models. Their best mean absolute error was 1.695 km/h. However, this work was performed on side road which had high pixel-to-meter ratio.

Fully apply of Deeplearning algorithm for vehicle speed estimation was presented in [19] and [20]. One-stage object detection algorithm YOLOv5 [37] was used to generate a series of vehicle detection box from side road video dataset. And another Deeplearning model was used to estimate speed of vehicle after trained. The average error was reported at 2.76 km/h and 4.08 km/h respectively.

### D. PROBLEMS AND SOLUTION

As our analysis, background-foreground vehicle detection technique faces unstable of background-foreground, changing of light intensity, and the detector detects other moving objects or shadows in the frame. These problems affect speed estimation accuracy. However, as our research, these problems could be solved by using Deeplearning-based vehicle detection algorithm. The algorithm has a good performance in


**FIGURE 1.** Vehicle detection and tracking methods. From left, (a) Background-foreground based [11] (Top is vehicle detection, bottom is centroid tracking), (b) Feature based [7] (License plate tracking), and (c) Machine learning based [17] (Box tracking).

vehicle detection under different situations such as nighttime, rainy, or cloudy [38]. In addition, because the algorithm detects by using the whole body of the vehicle, then, this is easy for humans to identify characteristics or classify types of the target vehicle for further purposes. Among various proposed Deeplearning-based vehicle detection algorithms, we select YOLOv3 as our vehicle detector for the following reasons:

1) YOLOv3 facilitated with multi-scale detection. This is very helpful for us because our target vehicle is moving away from the camera during the detecting and tracking period.

2) YOLOv3 has a fast processing speed. This is potentially allowed us for real-time processing.

3) YOLOv3 is well tested for vehicle detection in many studies.

4) YOLOv3 is user friendly [39] and its accuracy is acceptable in our long tracking period.

Although there are many variant versions of YOLO such v4, v5, R, x, PP-YOLO, v6, v7, v8. These different version were proposed with some improvement features for different kind of target application [40], [41], [42]. For example, YOLOv7 is the most accurate for tiny object detection due to higher input resolution [43] but, here, we do not need to detect a small vehicle.

Another problem with vision-based vehicle speed estimation is tracking accuracy. As in [17], tracking vehicle on the box given by the detector provides unstable tracking lines. Then, we propose to detect and track feature on the vehicle as [7]. We select DeepSORT for vehicle identification and we use pyramidal Lucas-Kanade optical flow algorithm for feature detection and tracking. In addition, we improve the speed estimation accuracy by applying vehicle movement pattern vector borrowing from [28].

## III. PROPOSED METHOD

### A. VEHICLE SPEED MEASUREMENT IN TRAFFIC VIDEO

In general, to get speed of the moving vehicle, we can simply measure by using two models. The first model is to measure traveling distance in a given unit of time and the second model is to measure vehicle traveling time in a given unit of distance. Then, speed in m/s or km/h can be calculated. Here, we are going to do both measurement models by using a traffic camera. However, to use a camera as a vehicle speed sensor, we have to note that the camera is a discrete sensor with a constant sampling rate at frame per seconds (fps). Therefore, the camera can only get a series of temporal movement location of the vehicle on the road surface in video frame. It is rather hard to detect exactly at a specific given point on the road. Then, the speed estimated by the camera is a discrete value.

For the first speed measurement model, measuring traveling distance of the vehicle in a given unit of time, we use camera frame rate as a given unit of time and measuring traveling distance of the vehicle between two consecutive frames. In this case, we have to note that error of the distance measurement is higher at the farther point comparing to the error measuring at the closest point from the camera due to pixel-to-meter ratio from the camera FoV. Hence, measuring traveling distance is better to be done at the closest point from the camera for the higher accuracy.

If we consider position of the vehicle between any two consecutive frames, traveling distance of the vehicle between these two frames called pixel displacement ($\Delta d$) is represented by:

$$\Delta d = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{1}$$

where $(x_i, y_i)$ is horizontal and vertical pixel position of the target vehicle on frame $i$ and $(x_{i+1}, y_{i+1})$ is the horizontal and vertical position of the target vehicle on the later frame $i + 1$. Speed of the vehicle ($v$) can be calculated by dividing the distance with traveling time which is camera frame rate (fps) as:

$$v = \frac{\Delta d}{fps} \tag{2}$$

Unit of $\Delta d$ is in pixel and fps is in seconds then $v$ is in pixel per seconds. If $\Delta d$ can be transferred to the distance in real world unit using pixel-to-meter ratio. Then, (2) can be used to estimate speed of moving vehicle between any two consecutive frames in m/s or km/h.

For the second speed measurement model, measuring traveling time in a given unit of distance, if we define a unit of distance by placing two virtual intrusion lines on the road surface in the video frame with distance $L$. Then, we count the number of frames that the vehicle takes to move from the first line to the second line ($n$) and multiply by the video frame rate (fps). Hence, the traveling time of the vehicle is measured. As mentioned earlier, camera is a discrete sensor then this traveling time is a discrete number given by $n \times$ fps where $n = 1, 2, 3, 4, \ldots, N$ and $n$ starts counting after the vehicle crosses the first virtual intrusion line until the vehicle crosses the second virtual intrusion line. Finally, speed of the vehicle can be calculated by:

$$v = \frac{L}{n \times fps} \tag{3}$$

where $L$ represents the distance in pixel unit and $v$ is the speed in pixel per seconds. If $L$ is defined in meter then the speed $v$ can be converted into km/h.

### B. VEHICLE MOVEMENT PATTERN AND SPEED ESTIMATION EQUATIONS

In this analysis phase, two speed measurement models in section A are implemented. We created some traffic videos by setting up a traffic camera at the height approximately 7.3 meters on a skycrossing bridge over Phaholyothin road, Pathumthani, Thailand as shown in Fig.2 (Google map coordinate: 14.07500439231682, 100.61762530378513). A new iPad Pro 2012 was installed as a traffic camera for capturing traffic videos. Its back camera was configured to Full HD resolution 1920 × 1080 pixels and its frame rate was 60 fps. Laser speed gun LTI 20/20 TruSpeed with accuracy +/− 2 km/h was used to measure ground truth speed of the target vehicle in the range between 40 to 60 meters from the camera. Voice reading was used to record speed value from the speed gun into the video during capturing period.

**FIGURE 2.** Traffic video recording and vehicle speed measurement on the skycrossing bridge over Phahonyothin road at the height 7.3m over the road. (Image on the road from Google street view.)

Based on our traffic video frame, four virtual intrusion lines Start line0, line1, line2, and line3 are defined at $y$ equals to 490, 310, 226, 164, respectively. Where the image pixel coordinate $x$=0 and $y$=0 is at the top-left corner of the image as shown in Fig.3. Distance among the lines are known based on the road mark sign referring to data from Thailand Department of Highways (DOH). The white dash line length in the green box on Fig.3 is 3 meters and the gap between each white dash lines is 9 meters. However, these data are only used as a reference, we do not use them in our speed estimation process. Our target vehicles are only the vehicles that pass the Region of Interest (ROI) as defined with the green box shown in Fig.3 because speed limit violation always occurs on this most right lane.

From Fig.3, we now can implement our two speed measurement models (2) and (3) by using four virtual intrusion lines. We track a target vehicle in ROI and then we calculate the pixel displacement $\Delta d$ together with the number of frames $n$ at each virtual intrusion line. A simple model of the vehicle moving in ROI is illustrated in Fig.4.

At this stage, two sets of temporal vehicle movement pattern can be introduced. First, a set of $\Delta d$ called ''Crossing

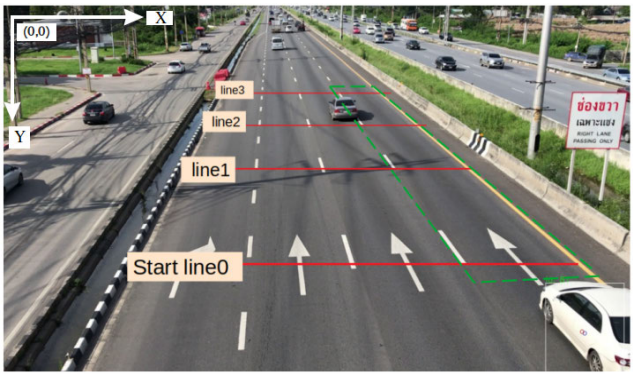


**FIGURE 3.** Four virtual intrusion lines (Red lines) and ROI location (Green dash box) are defined on the video frame size 1920 × 1080.

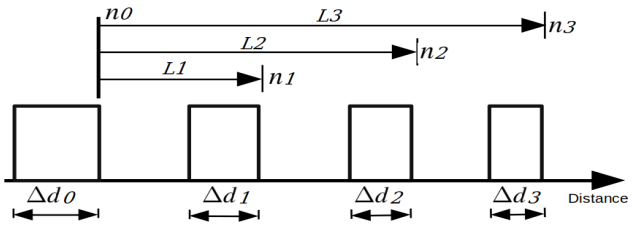


**FIGURE 4.** A simple model of a vehicle moving in ROI, $\Delta d0$, $\Delta d1$, $\Delta d2$ and $\Delta d3$ are the Crossing distance at each intrusion line. $n1$, $n2$, $n3$ are the number of frames that the vehicle takes to move in the distance $L1$, $L2$, $L3$, respectively.

distance''. It is a set of pixel displacements between two consecutive frames before and after the vehicle crosses each virtual intrusion line at speed $v$. Crossing distance represents by:

$$\Delta d_v = [\Delta d0_v, \Delta d1_v, \Delta d2_v, \Delta d3_v] \tag{4}$$

where $\Delta d_v$ is a Crossing distance belong to the vehicle moving at speed $v$ km/h in ROI. $\Delta d0_v$, $\Delta d1_v$, $\Delta d2_v$, and $\Delta d3_v$ are the pixel displacement at line0, line1, line2, and line3, respectively.

The second set of the temporal vehicle movement pattern is a set of the number of frames $n$ called ''Frame counter number''. It is a set of the number of frames that the vehicle takes after crossing line0 until the vehicle crosses the other lines at speed $v$. Frame counter number represents by:

$$n_v = [n0_v, n1_v, n2_v, n3_v] \tag{5}$$

where $n_v$ is a Frame counter number belong to the vehicle moving at speed $v$ km/h in ROI. $n1_v$, $n2_v$, and $n3_v$ are the number of frames that the vehicle took after crossing line0 until the vehicle crosses line1, line2, and line3, respectively. $n0_v$ is the starting frame and it always equals to one.

In our assumption, if the target vehicle moves at a constant speed $v_c$ km/h in ROI and we know the relationship between image and real-world coordinate, then, speed of the target vehicle can be calculated with (2) by using any $\Delta d$ in (4) or it can be calculated with (3) by using $n1$, $n2$ or $n3$ in (5). All calculated speeds $v$ have to equal $v_c$. We can see that the speed

of the vehicle moving in the video frame can be calculated easily if the vehicle is tracked precisely to generate Crossing distance and Frame counter number.

In this study, we implement a Deeplearning-based object detection, YOLOv3, to detect position of vehicle in the video frame. The detected vehicle is passed as a box with (*x, y, width, height*) to DeepSORT to get an identification number (ID). Then, DeepSORT tracks the vehicle throughout the ROI. However, the box given by YOLOv3 is not accurate enough to reflect vehicle trajectory (similar to Fig.1-c), then, GoodFeatureToTrack and Pyramidal Lucas-Kanade optical flow algorithms are implemented to get better tracking accuracy on the vehicle trajectory. At this study stage, we manually point GoodFeatureToTrack algorithm to detect features at the wheel area of the vehicle. Tracking features at the wheel area, which is the closest part of the vehicle to the road surface, helps to reduce speed estimation error caused by the height of the tracking point on the vehicle in 3D world coordinate. After some features on the wheel area are selected, Pyramidal Lucas-Kanade optical flow algorithm is applied to track all the selected features. The centroid of the selected features is used as a reference point to generate vehicle tracking data.

Fig.5 shows six features on the wheel area of the vehicle selected by GoodFeatureToTrack (Red dots). Their centroid (Blue dot) is calculated as the reference point to generate vehicle tracking data. The tracking result is illustrated in Fig.6. We compare the tracking data generated by using GoodFeatureToTrack and Pyramidal Lucas-Kanade optical flow algorithm (The wheel area tracking as Blue dot line) and the tracking data generated by using YOLOv3 (Center of the box tracking as Green dot line).

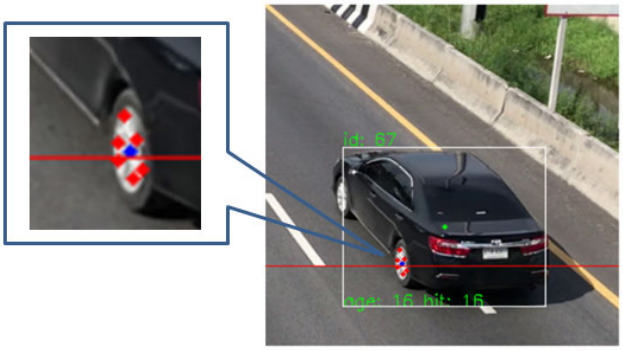

**FIGURE 5.** Six features on the wheel area are selected by GoodFeatureToTrack algoritm (Red dots) and their centroid (Blue dot) is calculated as the reference point to generate vehicle tracking data.

In this study step, we analyze 187 traffic videos recorded by the method mentioned previously. In the video, we have only one target vehicle in one video file, each video is processed manually to get the vehicle tracking data. The process consists of three manual steps as follows:

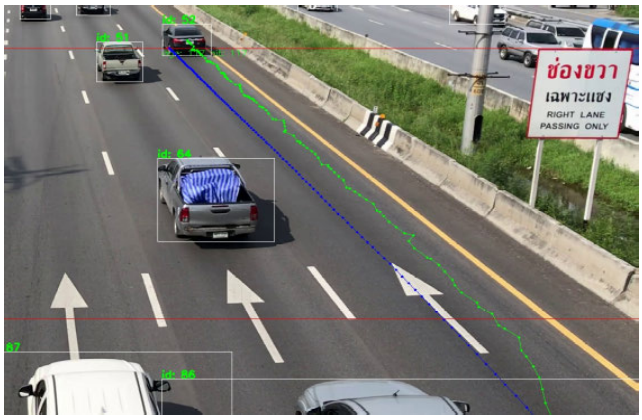


**FIGURE 6.** The better accuracy of vehicle tracking data provided by GoodFeatureToTrack and Pyramidal Lucas-Kanade optical flow algorithm at the wheel area (Blue dot line) compared to the box tracking data provided by YOLOv3 (Green dot line).

1) We create a Python script with the implementation of YOLOv3 and DeepSORT to generate the vehicle identification number (ID) of every vehicle that entering the ROI in each video file. Results from the script are the image of every vehicle entering ROI with its ID.

2) Virtual inspection is performed on each video file to identify the target vehicle and get its speed from voice recording in the video. Then, the target vehicle is matched to the image from the first step. Here, we know the ID and speed of each target vehicle. This information is manually entered into a CSV (Comma Separate Values) file including the video file name.

3) We create a new Python script with the implementation of YOLOv3, DeepSORT, GoodFeatureToTrack, and Pyramidal Lucas-Kanade optical flow algorithm. The script takes the video file name and the ID from the CSV file and searches the target vehicle in the video file. When the target vehicle is found, the script makes a pause and allows us to point GoodFeatureToTrack to the wheel area of the target vehicle. Then, it continues to use Pyramidal Lucas-Kanade optical flow algorithm to track the selected features until the end of ROI (Blue dot line in Fig.6).

Finally, from the third step, we get only 135 valid tracking data of 135 target vehicles. The data include the pixel position of the centroid and its frame number. Some tracking data are not valid because the tracking point disappears during the tracking period, the target vehicle changes lane, and no features are selected by GoodFeatureToTrack algorithm.

Fig.7 shows tracking data of 9 target vehicles at ground truth speed 72, 77, 82, 87, 92, 97, 102, 108, and 112 km/h, respectively (From left to right). In the Figure, we can find a linear decrease in Traveling time (frameCount on the plot) when the target vehicle moves at a higher speed (See the bottom black arrow). In contrast, $\Delta d$ (pixel displacement) increases by increasing the speed (The top blue arrow).

From the tracking data of 9 target vehicles in Fig.7, we apply the intrusion lines onto the data to get Crossing

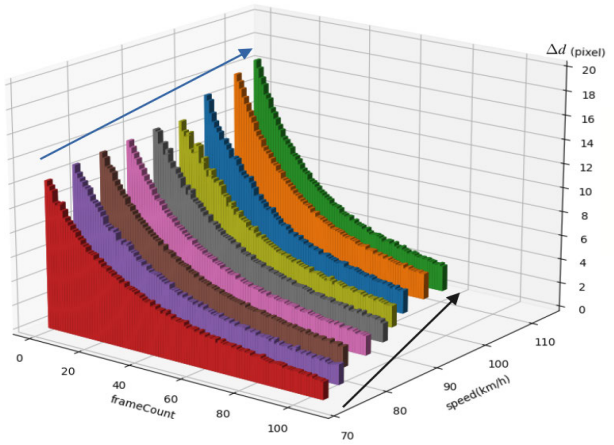


**FIGURE 7.** Increasing the speed causes a decrease in Traveling time (frameCount) in ROI (Bottom black arrow) while the pixel displacement increases by increasing the speed (Top blue arrow).

distance at each speed (4). Then, we plot the Crossing distance of 9 target vehicles comparing to their speed as shown in Fig.8. The plot shows better visibility to see the linear relationship between vehicle speed and Crossing distance at each virtual intrusion line. We can see the increasing speed causes an increase in Crossing distance.

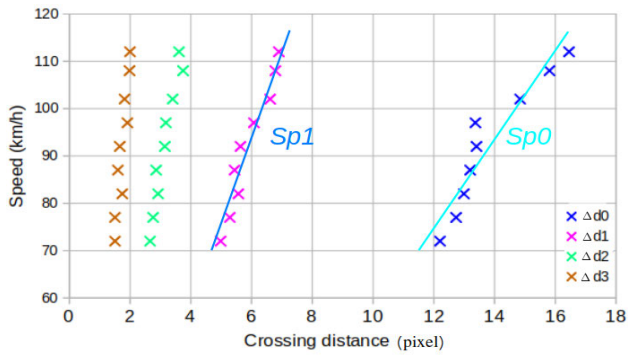


**FIGURE 8.** Relationship between Crossing distance ($\Delta d0$, $\Delta d1$, $\Delta d2$, and $\Delta d3$) and vehicle speed.

From the Crossing distance data in Fig.8, The linear relationship equation between Crossing distance and vehicle speed on the virtual intrusion line0 and line1 can be computed by:

$$Sp_0 = (slope_0 \times d_0) + B_0 \tag{6}$$

and,

$$Sp_1 = (slope_1 \times d_1) + B_1 \tag{7}$$

where $slope_0$, $B_0$ and $slope_1$, $B_1$ are a constant value that can be calculated by using the linear curve fitting method.

In the similar way, from the tracking data of 9 target vehicles in Fig.7, we again apply the intrusion lines onto the data to get Frame counter number at each speed (5). Then, we plot the Frame counter number of 9 target vehicles comparing to their speed as shown in Fig.9. The plot shows better visibility to see the linear relationship between vehicle speed and Frame counter number at each virtual intrusion line. We can see the increasing speed causes an decrease in Frame counter number.

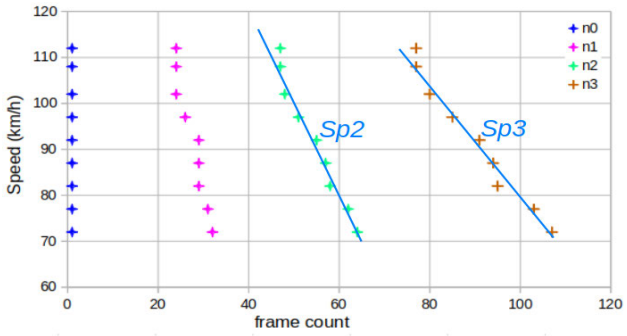


**FIGURE 9.** Relationship between Frame counter number at each virtual intrusion line ($n0$ $\mathcal{D}$ $1$, $n1$, $n2$, and $n3$) and vehicle speed.

From the Frame counter number data in Fig.9, The linear relationship equation between Frame counter number and vehicle speed on the virtual intrusion *line*2 and *line*3 can be computed by:

$$Sp_2 = (slope_2 \times n_2) + B_2 \tag{8}$$

and,

$$Sp_3 = (slope_3 \times n_3) + B_3 \tag{9}$$

where $slope_2$, $B_2$ and $slope_3$, $B_3$ are a constant value that can be calculated by using the linear curve fitting method.

## IV. EXPERIMENTS

### A. DATASET

The similar process to create the dataset in the analysis phase is performed in this phase but the iPad camera is replaced by iPhone 13 Mini camera. The back camera of iPhone 13 Mini is configured to Full HD resolution 1920 × 1080 pixels at 120 fps. The same laser speed gun is used to measure speed of the target vehicle. The voice reading recording is used to record the measured speed value into the video. The same location on the skycrossing bridge is used to create the new dataset. The videos are captured in the afternoon of December $1^{st}$, 2022. The captured time is between 11:00 to 16:00 Local time. The weather conditions on the capturing day are clear with very bright sunlight. All target vehicles are on the most right lane in the video frame. We capture it as a short video file with a few target vehicles in the video for easy processing in the next step.

From 138 video files, with 898 target vehicles, the steps to process the video file are similar to our analysis phase. But one video file now can consist of more than one target vehicle. Then, the frame number when the target vehicle enters the camera FoV has to be added to the CSV file for reference. Information in the CSV file includes the video file name, the entering frame number, the vehicle ID, and the ground truth

speed. The total 898 target vehicles have a minimum speed at 44 km/h and a maximum speed at 127 km/h. There are 70% of the target vehicles moving at speed between 70 to 90 km/h.

The whole dataset here are available online for research purposes and shall be considered as one of our work contribution.(https://drive.google.com/drive/folders/12F7AlJiv2AMiJ1DZ4PiOlAZlEXzSUPS7?usp=drive_link)

### B. EXPERIMENTAL SETTING

In our analysis phase, the wheel of the target vehicle was pointed manually for feature selection but this is rather hard for the new dataset. Then, we train one YOLOv3 model especially for wheel detection. Training data are taken from the vehicle entering the ROI in the analysis phase.

Another change here is we track directly to the selected features on the wheel area because tracking by the centroid was not stable. Here, two tracking points are selected among many good features given by GoodFeatureToTrack algorithm. Flow diagram of our proposed system illustrates in Fig.10.

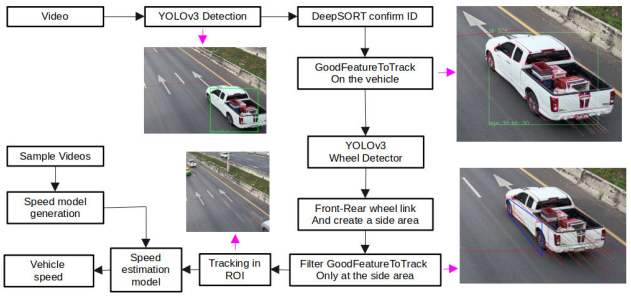


**FIGURE 10.** Flow diagram of our system and example of outputs.

From the camera view of the new dataset, first, four visual intrusion lines including line0, line1, line2 and line3 are defined on the video frame at y = 655, 447, 351, and 278, respectively (The top-left corner is $x$=0 and $y$=0 as in Fig.3). Second, we create a new CSV file from the new dataset by following step 1 and step 2 in the analysis phase. At this time, we have to add the frame number that the target vehicle enters the camera view. Last, we create a new Python script to implement YOLOv3 for vehicle detection, DeepSORT, YOLOv3 for wheel detection, GoodFeatureToTrack, and Pyramidal Lucas-Kanade optical flow. The workflow of the script is presented in Fig.10.

Referring to the flow diagram, after entering the camera view, all vehicles are detected by YOLOv3 and receive the ID from DeepSORT in a few frames later. Then, we identify the target vehicle by using the ID in the CSV file and the target vehicle should appear at the corresponding frame number. Then, GoodFeatureToTrack is applied to the target vehicle to get tracking features (All big red dots over the white vehicle in Fig.10 or more clear in Fig.11). We track every selected features on the target vehicle until the vehicle enters wheel detection area (at y=750). At this point, YOLOv3 for wheel detection is applied to the vehicle to identify location of the wheels (Two red rectangles at the wheels in Fig.11). Then, the blue rectangle is created to link between the box of two wheels (Fig.11). Later, only two tracking features that are nearest to the horizontal half of the blue rectangle will be selected. From now on, these two selected features will be tracked until the end of the ROI (Fig.12). The pixel position of the two selected features on the target vehicle in each video frame will be recorded together with its frame number as a tracking data. Fig.12 shows two tracking features at the front wheel are tracked in the ROI.

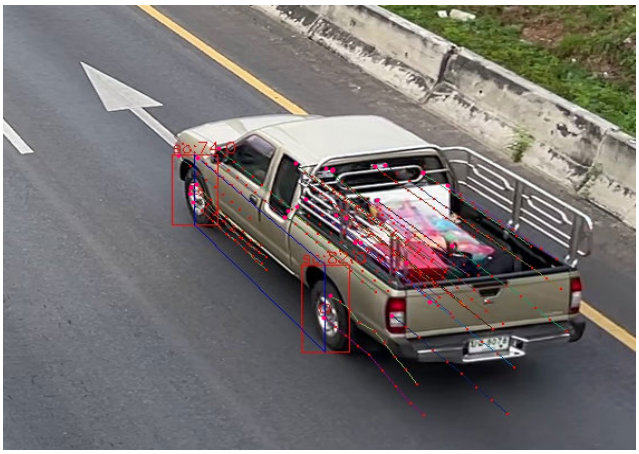

**FIGURE 11.** Wheel detection (Red boxes) and area selection (Blue box) at the side of target vehicle after the vehicle crossing wheel detection point at y=750.

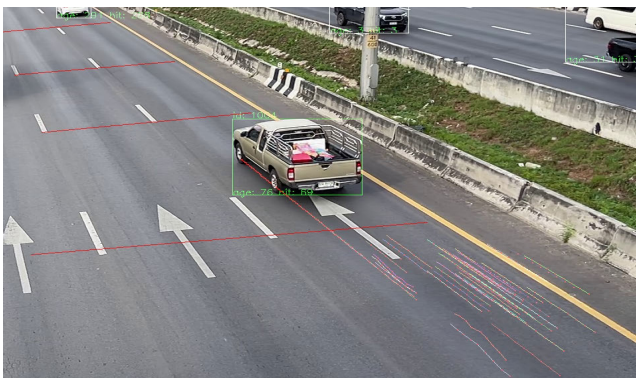

**FIGURE 12.** Two selected features at the front wheel area are continuously tracked after the target vehicle enters the region of interest.

From 898 tracking data of 898 target vehicles, we use vanishing point technique to remove tracking data that the target vehicle is moving out from the most right lane and the tracking features are not on the target vehicle. The other invalid tracking data are also removed including tracking data that lose tracking features and the vehicle ID changes and it can not recover by DeepSORT. Finally, we have only 833 valid tracking data. Speed distribution of the valid tracking data are illustrated in Fig.13.

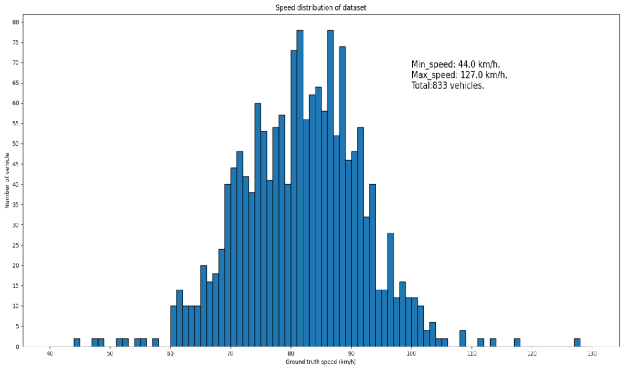


**FIGURE 13.** The number of vehicles at each ground truth speed in the dataset after filtering invalid data.

For the video processing and computation in the analysis phase, we worked on the work station CPU: Intel® Core i7-8750H 2.2GHz with memory 32 GB and GPU: nVidia GeForce GTX 1060 Mobile. The operating system was Ubuntu 18.04; processing software was coded by using Python 3.6.9 with supported libraries from OpenCV 4.2.0, CUDA 10.0 CuDNN 7.6, and Tensorflow 1.13.1 for video processing. But the experiment is performed on the difference hardware configuration. It is on CPU: Intel® Core i7-9700K 3.6GHz with memory 16 GB and GPU: nVidia GeForce GTX 1080Ti. The operating system is Ubuntu 18.04; Processing software is Python 3.6.9, OpenCV 4.4.0, CUDA 11.0, CuDNN 8.0, Tensorflow 2.6.0 for the implementation of Multilayer Perceptron (MLP) and the testing is performed on the same hardware.

### C. EVALUATION

Statistical evaluation Mean Absolute Error (MAE) is adopted as an evaluation tool. It is the summation of each absolute error from individual estimated speed then divided by the number of vehicles. MAE is defined as:

$$MAE = \frac{1}{M}\sum_{m=1}^{m=M} \mid v_m^g - v_m^e \mid \tag{10}$$

where $M$ is the number of vehicles to be measured, $v_m^g$ is the ground truth speed of vehicle $m$, and $v_m^e$ is the estimated speed of vehicle $m$.

Root-Mean-Square Error (RMSE) is used to represent the sensitivity of variation to large errors. It is the square root of the mean of squared residual defined as:

$$RMSE = \sqrt{\frac{\sum_{m=1}^{m=M}\left(v_m^g - v_m^e\right)^2}{M}} \tag{11}$$

### D. SPEED ESTIMATION RESULT

Results from the experiment are the estimated speed by using our metrics model as an input of the linear equations. Neural Network speed estimation is a parallel test for comparing accuracy but the number of testing data is lower than the linear estimation equation method.

#### 1) LINEAR EQUATIONS SPEED ESTIMATION

We test the vehicle tracking data with 3 levels of frame rate including 120fps, 60fps, and 30fps. From the original frame rate at 120 fps, the tracking data of 20 vehicles are selected as a sample to find *slope* and *B* of each linear equation (6), (7), (8), and (9).

The sample with five tracking data at speed 79, 86, 93, and 100 km/h are processed by Framedrop (1:1). Then, they are filtered by using (4) and (5) to get Crossing distance and Frame counter number at each speed. Later, the linear curve fitting is applied to get *slope* and *B* for each linear equation. Finally, the equations are used to estimate the speed of 813 vehicles. The computation flow is illustrated in Fig.14. The same computation process is repeated but Framedrop is changed to 2:1 and 4:1 to drop the tracking data down to 60fps and 30fps respectively.

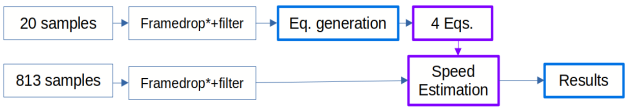


**FIGURE 14.** Computation flow for the experiment, Framedrop is 1:1, 2:1, and 4:1.

Table 1 shows the linear equation (6), (7), (8), and (9) with their *slope* and *B* at the difference Framedrop (fps) received from the sample.

**TABLE 1.** The constant value *slope* and *B* of each linear equation received from the sample 20 tracking data.

| Speed Eq. | $fps$ | slope | B | $R^2$ |
|---|---|---|---|---|
| *Sp0* | 120 | 9.4513 | 3.5 | 0.9937 |
| | 60 | 4.7482 | 3.5 | 0.9872 |
| | 30 | 2.3535 | 3.5 | 0.9809 |
| *Sp1* | 120 | 22.9307 | 3.0 | 0.9948 |
| | 60 | 11.5854 | 3.0 | 0.9958 |
| | 30 | 5.7692 | 3.0 | 0.9951 |
| *Sp2* | 120 | -0.6711 | 160.0 | 0.9970 |
| | 60 | -1.3284 | 160.0 | 0.9971 |
| | 30 | -2.5964 | 160.0 | 0.9960 |
| *Sp3* | 120 | -0.4194 | 163.0 | 0.9973 |
| | 60 | -0.8317 | 163.0 | 0.9974 |
| | 30 | -1.6343 | 163.0 | 0.9962 |

Tables 2, 3, and 4 show speed estimation evaluation results at each fps, 120fps, 60fps, and 30fps, respectively. The estimation uses (6), (7), (8), and (9) with their *slop* and *B* in Table 1.

Based on MAE and RMSE values, the best accuracy of speed estimation using Crossing distance equation (6) and (7) can be found in Table 4 at 30fps. The best of (6) has MAE 5.92 km/h and RMSE 12.42 km/h while the best of (7) has MAE 4.37 and RMSE 7.65 km/h. The Crossing distance equations give lower accuracy when applying with 60fps and 120fps as shown in Table 2 and Table 2 respectively.

**TABLE 2.** Estimated speed statistical evaluation of 813 vehicles by using $Sp_0$, $Sp_1$, $Sp_2$, and $Sp_3$ at 120*fps*.

| Speed Eq. | MAE(km/h) | RMSE(km/h) |
|---|---|---|
| $Sp_0$ | 10.98 | 33.63 |
| $Sp_1$ | 7.37 | 21.98 |
| $Sp_2$ | 3.38 | 4.69 |
| $Sp_3$ | 3.51 | 6.72 |

**TABLE 3.** Estimated speed statistical evaluation of 813 vehicles by using $Sp_0$, $Sp_1$, $Sp_2$, and $Sp_3$ at 60*fps*.

| Speed Eq. | MAE(km/h) | RMSE(km/h) |
|---|---|---|
| $Sp_0$ | 6.74 | 17.56 |
| $Sp_1$ | 5.03 | 12.23 |
| $Sp_2$ | 3.80 | 5.59 |
| $Sp_3$ | 4.03 | 7.70 |

**TABLE 4.** Estimated speed statistical evaluation of 813 vehicles by using $Sp_0$, $Sp_1$, $Sp_2$, and $Sp_3$. at 30*fps*.

| Speed Eq. | MAE(km/h) | RMSE(km/h) |
|---|---|---|
| $Sp_0$ | 5.92 | 12.42 |
| $Sp_1$ | 4.37 | 7.65 |
| $Sp_2$ | 5.30 | 8.05 |
| $Sp_3$ | 5.58 | 11.17 |

The better speed estimation accuracy can be found in Table 2 by using Frame counter number equation (8) and (9). The accuracy of (8) is with MAE 3.38 km/h and RMSE 4.69 km/h. It is a bit higher than the accuracy of (9) with MAE 3.51 km/h and RMSE 6.72 km/h. However, the accuracy of (9) at 120fps is still better than using Crossing distance equations at all video frame rates. Frame counter number equations are opposite to Crossing distance equations. For Frame counter number equations, their accuracy increases at the higher video frame rate (fps) as shown in Table 2, Table 3 and Table 4.

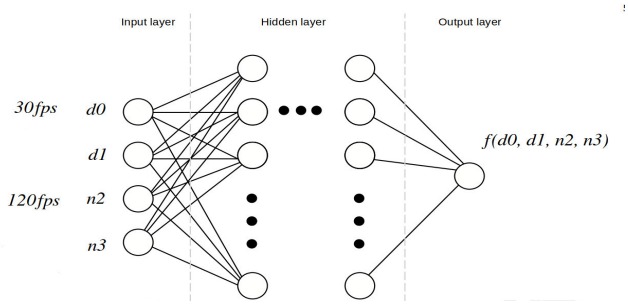


**FIGURE 15.** Inputs and output of the Fully connected neural network model for the experiment.

### 2) NEURAL NETWORK SPEED ESTIMATION

We test if a fully connected neural network can get the same accuracy for vehicle speed estimation after being trained by some of our data. We create 5 hidden layers of 20 Dense nodes. The Activation function is the combination of tanh and linear. The dataset is the same 813 tracking data but we take only Crossing distance data at 30fps and Frame counter number at 120fps because they can provide the best accuracy. The output of the model is the estimated speed of the target vehicle which is a function of *d0, d1* at 30fps and *n2, n3* at 120fps as presented in Fig. 15.

The tracking data are separated into training data and testing data. The training data are about 70% of the dataset (570 vehicles) and the remaining 30% (243 vehicles) are used for speed estimation evaluation of the model. Fig.16 shows speed distribution of the training data and Fig.17 shows speed distribution of the testing data.

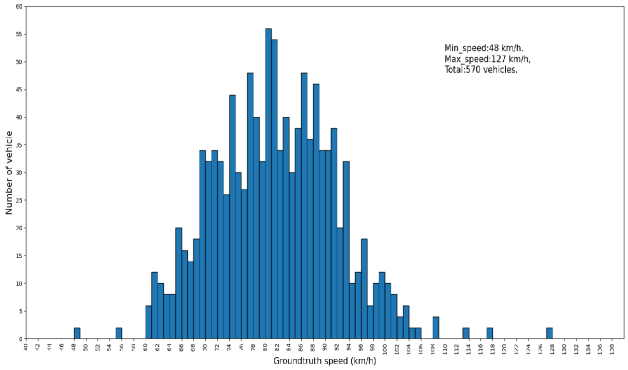


**FIGURE 16.** Speed distribution of 570 vehicles from training dataset.

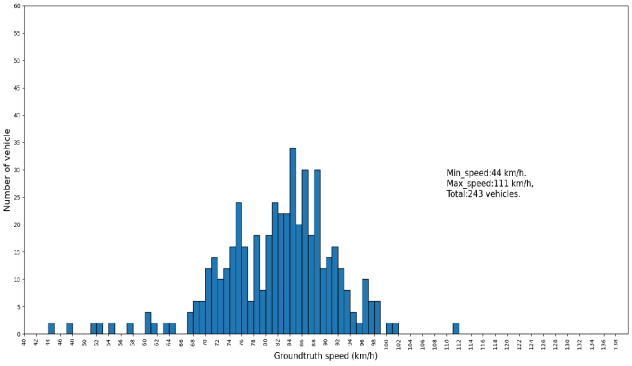


**FIGURE 17.** Speed distribution of 243 vehicles from test dataset.

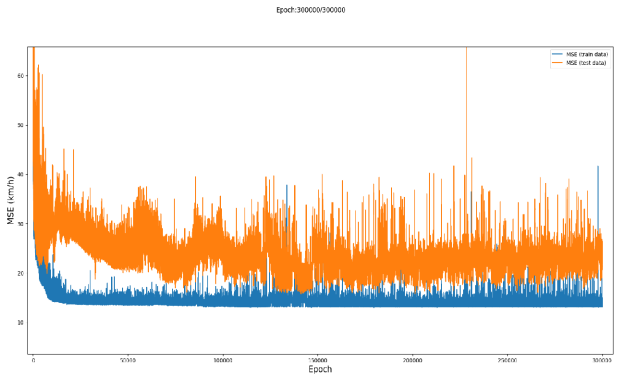


**FIGURE 18.** Loss of training data and testing data during training upto 300000 epoch.

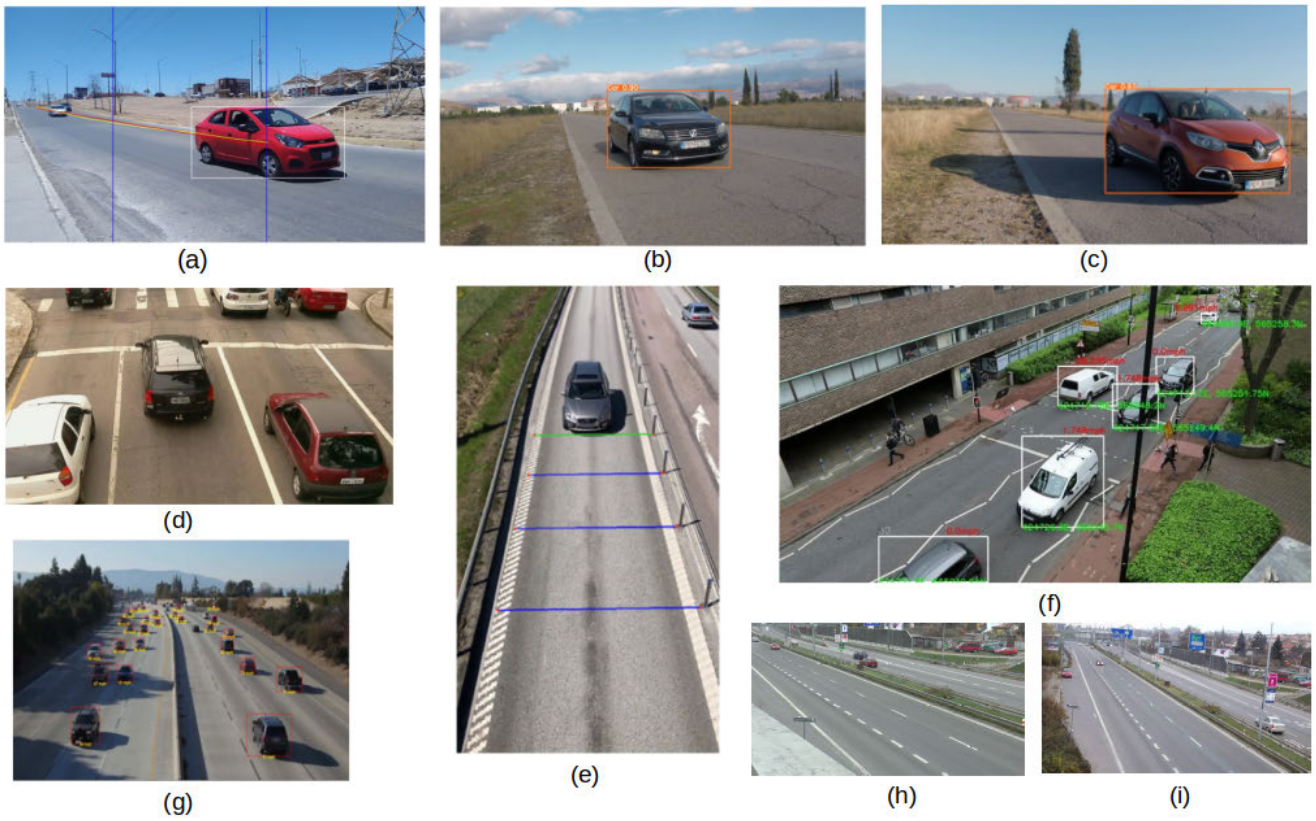


**FIGURE 19.** FoV comparison among each study, (a) [18], (b) [19], (c) [20], (d) [7], (e) [28], (f) [17], (g) [33], (h) [44], (i) [16].

Mean Squared Error (MSE) is configured as the loss for training the model. Fig. 18 is the comparison between loss of training data and testing data. The best result is at Epoch 141,197 with MAE 3.07 km/h and RMSE 3.98 km/h.

To compare speed estimation accuracy using the neural network model with the accuracy using the linear equations, the same testing data (243 vehicles) are statically summarized in Table 5. This shows that the neural network model provides better performance than the linear equations when using our metrics as the input.

**TABLE 5.** Estimated speed statistical evaluation of the testing data 243 vehicles by $Sp_0$, $Sp_1$ at 30fps, $Sp_2$, $Sp_3$ at 120fps.

| Speed Eq. | MAE(km/h) | RMSE(km/h) |
|---|---|---|
| $Sp_0$ | 5.09 | 9.83 |
| $Sp_1$ | 3.81 | 4.94 |
| $Sp_2$ | 3.75 | 5.56 |
| $Sp_3$ | 3.60 | 5.39 |

### 3) EXPECTED RESULT

In comparison to other research, many different techniques are applied to video-based speed measurement as discussed in [5]. Here, we compare some similar techniques for accuracy comparison based on their report as shown in Table 6.

The first four studies, in Table 6, used a high pixel-to-meter ratio method (see Fig.19 (a),(b),(c)). In [18], their accuracy can reach MAE 0.95 km/h by using linear estimation method while Multilayer Perceptron can do at 3.17 km/h. Our Multilayer Perceptron model shows better accuracy with our input metric. We can reach a lower MAE at 3.07 km/h and this is also agreed by its RMSE at 3.98 km/h. Our linear equation has a higher error because we have a lower pixel-to-meter ratio by using the farther FoV. In another study [19], their speed estimation by using 1D-CNN model can do better than ours at RMSE 2.76 km/h but this is not too far from ours at 3.98 km/h. Although, they worked with a high pixel-to-meter ratio. Our Multilayer Perceptron model is also a little bit better in accuracy comparing to RNN speed estimation model [20]. They reported at RMSE 4.08 km/h. However, we have to note that in [20], they worked on side-road FoV with a high pixel-to-meter ratio video frame.

Luvizon et al. [7] implemented KLT tracking on the license plate of the vehicle with a top-down near FoV camera. Their reported errors of speed estimation were in between −4.68 km/h and +6.00 km/h but most of the vehicles had a low speed between 10 to 59 km/h.

Javadi et al. [28] reported their speed estimation average error 1.77% only at the speed of vehicle at 72, 73.8, 91.08, and 94.32 km/h.

**TABLE 6. Comparison of speed estimation between start-of-the-art works and our proposed method.**

| Author | Method | | Estimation algorithm | MAE | RMSE | Conditions |
|---|---|---|---|---|---|---|
| | Detection | Tracking | | | | |
| Héct. [18] | YOLOv3 | Kalman filter | Linear regression | 0.95 km/h | - | Zoom-side road at one vehicle. |
| Héct. [18] | YOLOv3 | Kalman filter | Multilayer Perceptron | 3.17 km/h | - | Zoom-side road at one vehicle. |
| Cvij. [19] | YOLOv5 | - | 1D-CNN | - | 2.76 km/h | Zoom-side road at one vehicle. |
| Peru. [20] | YOLOv5 | - | RNN | - | 4.08 km/h | Zoom-side road at one vehicle. |
| Luvi. [7] | Background-Foreground | KLT | Linear model | -0.50 km/h | - | Top-down near FoV, Low speed 10 to 69 km/h. |
| Java. [28] | Moving pattern vectors | Intrusion lines | PDF of moving pattern vectors | 1.77% | | Top-down near Fov. |
| Bell. [17] | YOLOv2 | SORT | Camera perspective | 1.85 km/h | 2.25 km/h | Top-down side near FoV, Low speed 1.6 to 15.8 km/h. |
| Amit. [33] | Mask-RCNN | SORT, DeepSORT | - | - | 15.35 km/h | Similar to ours |
| Dong. [44] | 3D ConvNets | Optical flow | 3D ConvNets | 2.71 km/h | 14.62 km/h | Similar to ours |
| Soch. [16] | Faster-RCNN | Kalman filter | Linear model | 1.10 km/h | - | Similar to ours, 3D model metadata of vehicles |
| Our | YOLOv3 | DeepSORT | Linear model | 3.38 km/h | 4.69 km/h | Far FoV. |
| Our | YOLOv3 | DeepSORT | Multilyer Perceptron | 3.07 km/h | 3.98 km/h | Far FoV. |

Bell et al. [17] used SORT to track YOLOv2 detection box. Their approach presented promising performance of the computation processes with an average RMSE at 2.25 km/h (0.625 m/s). However, their experiment was performed on urban road traffic under non-free-flowing conditions with a few samples of vehicles.

Kumar et al. [33], similar FoV and method with ours, used Mask-RCNN for vehicle detection and tracking by SORT in the NVIDIA AI City challenge 2018. Their speed estimation was reported at RMSE 15.35 km/h (9.54 mph) which has a lower accuracy than ours.

Dong et al. [44] can reach a better MAE 2.71 km/h than ours by using 3D ConvNets. But their RMSE 14.62 km/h shows the variation of large error which is similar to [33].

It clearly illustrates that our input metrics of the Multilayer Perceptron model have a better performance when comparing our RMSE to RMSE reported by [33] and [44]. However, we can not reach to accuracy of Sochor et al. [16] at MAE 1.10 km/h. But their method requires 3D model metadata of vehicles while we use only simple metric model from only 20 vehicles. The comparison results are illustrated in Table 6 and Fig.19 shows the difference of FoV between each study.

## V. CONCLUSION

A simple video-based vehicle speed measurement method is presented in this study. The main motivation is to integrate a road traffic camera system with an automatic vehicle speed limit enforcement function.

Two speed measurement models are considered here, including measuring traveling distance of the moving vehicle in a given unit of time (Crossing distance) and measuring traveling time of the moving vehicle in a given unit of distance (Frame counter number). To implement the measurement models in the traffic video, four virtual intrusion lines are defined in the Region of Interest (ROI). Then, YOLOv3, DeepSORT, GoodFeatureToTrack, and Pyramidal Lucas-Kanade optical flow algorithm are implemented together to detect and track the target vehicle at the wheel area in the ROI. The tracking data of 833 target vehicles are analyzed based on two measurement models. Tracking data of 20 vehicles at the ground truth speed 72, 77, 82, 87, 92, 97, 102, 108, and 112 km/h are selected to create the speed estimation metrics that are used to estimate speed of 813 vehicles. The best speed estimation accuracy from the results is with MAE 3.38 km/h RMSE 4.69 km/h. This best accuracy is received from the relationship between Frame counter number and the ground truth speed at the third virtual intrusion line (line2) by using video frame rate at 120fps. The accuracy is lower to be MAE 3.80 km/h RMSE 5.59 km/h and MAE 5.30 km/h RMSE 8.05 km/h when the frame rate is dropped down to 60fps and 30fps respectively. While our Multilayer Perceptron model provides the best accuracy with MAE 3.07 km/h RMSE 3.98 km/h. The model consists of 20 Denses 5 hidden layers and it is trained by 70% of dataset. The remaining 30% is used to evaluate accuracy of the model.

Our proposed speed measurement metrics have a good accuracy compared to the previous study and the metrics are simple to create with only 20 samples at a wide range speed. The metrics can also be applied as the input of Multilayer Perceptron model for speed estimation. However, our method still has some limitations as follows. The first one is the failure of vehicle detection and wheel tracking because of the occlusion from a vehicle in the next lane. The worst case happens when there is a big vehicle such as a truck or a bus in the next lane. These big vehicles will cover all camera views. The second problem is that even though our approach does not mainly rely on the vehicle detector. Missing detection of the vehicle is allowed for a few frames. However, if the missing detection happens during the target vehicle crosses the intrusion line, this will result in a large error in speed estimation. For future work, we plan to improve the speed measurement of the dataset with a higher accuracy of the ground truth speed.

## REFERENCES


[1] T. V. Mathew. (2023). *Automated Traffic Measurement*. Indian Institute of Technology, Bombay, India. Accessed: Sep. 11, 2023. [Online]. Available: https://www.civil.iitb.ac.in/~vmtom/nptel/524_AutoMer/web/web.html

[2] P. Michalaki, M. Quddus, D. Pitfield, M. Mageean, and A. Huetson, "A sensor-based system for monitoring hard-shoulder incursions: Review of technologies and selection criteria," in *Proc. 5th Int. Conf. Transp. Traffic Eng.*, vol. 81, Oct. 2016, pp. 1–8.

[3] U.S. Department of Transportation Federal Highways Administration. (2014). *Traffic Monitoring Guide eBook*. Washington, DC, USA. Accessed: Sep. 11, 2023. [Online]. Available: https://www.fhwa.dot.gov/policyinformation/tmguide/tmg_2013/traffic-monitoring-theory.cfm

[4] H. A. Kurdi, "Review of closed circuit television (CCTV) techniques for vehicles traffic management," *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, no. 2, pp. 199–206, Apr. 2014, doi: 10.5121/IJCSIT.2014.6216.

[5] D. Fernández Llorca, A. Hernández Martínez, and I. García Daza, "Vision-based vehicle speed estimation: A survey," *IET Intell. Transp. Syst.*, vol. 15, no. 8, pp. 987–1005, Aug. 2021, doi: 10.1049/ITR2.12079.

[6] A. G. Yabo, S. I. Arroyo, F. Safar, and D. Oliva, "Vehicle classification and speed estimation using computer vision techniques," in *Proc. Conf. Argentina Assoc. Control Automat. (AADECA)*, Buenos Aires, Argentina, 2016, pp. 1–6. [Online]. Available: https://core.ac.uk/download/pdf/296389522.pdf

[7] D. C. Luvizon, B. T. Nassu, and R. Minetto, "A video-based system for vehicle speed measurement in urban roadways," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1393–1404, Jun. 2017, doi: 10.1109/TITS.2016.2606369.

[8] C. Wang and A. Musaev, "Preliminary research on vehicle speed detection using traffic cameras," in *Proc. IEEE Int. Conf. Big Data*, Los Angeles, CA, USA, Dec. 2019, pp. 3820–3823, doi: 10.1109/BigData47090.2019.9006233.

[9] W.-P. Wu, Y.-C. Wu, C.-C. Hsu, J.-S. Leu, and J.-T. Wang, "Design and implementation of vehicle speed estimation using road marking-based perspective transformation," in *Proc. IEEE 93rd Veh. Technol. Conf.*, Helsinki, Finland, Apr. 2021, pp. 1–5, doi: 10.1109/VTC2021-Spring51267.2021.9448813.

[10] S. S. Wardha, S. M. Deokar, S. S. Patankar, and J. V. Kulkarni, "Development of automated technique for vehicle speed estimation and tracking in video stream," in *Proc. 2nd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*, Bangalore, India, May 2017, pp. 940–944, doi: 10.1109/RTEICT.2017.8256736.

[11] G. Cheng, Y. Guo, X. Cheng, D. Wang, and J. Zhao, "Real-time detection of vehicle speed based on video image," in *Proc. 12th Int. Conf. Measuring Technol. Mechatronics Autom. (ICMTMA)*, Phuket, Thailand, Feb. 2020, pp. 313–317, doi: 10.1109/ICMTMA50254.2020.00076.

[12] B. Krishnakumar, K. Kousalya, R. S. Mohana, E. K. Vellingiriraj, K. Maniprasanth, and E. Krishnakumar, "Detection of vehicle speeding violation using video processing techniques," in *Proc. Int. Conf. Comput. Commun. Informat. (ICCCI)*, Coimbatore, India, Jan. 2022, pp. 1–7, doi: 10.1109/ICCCI54379.2022.9740909.

[13] W. Wu, V. Kozitsky, M. E. Hoover, R. Loce, and D. M. T. Jackson, "Vehicle speed estimation using a monocular camera," *Proc. SPIE*, vol. 9407, pp. 17–30, Mar. 2015, doi: 10.1117/12.2083394.

[14] A. E. Bouziady, R. O. H. Thami, M. Ghogho, O. Bourja, and S. E. Fkihi, "Vehicle speed estimation using extracted SURF features from stereo images," in *Proc. Int. Conf. Intell. Syst. Comput. Vis. (ISCV)*, Fez, Morocco, Apr. 2018, pp. 1–6, doi: 10.1109/ISACV.2018.8354040.

[15] C. Liu, D. Q. Huynh, Y. Sun, M. Reynolds, and S. Atkinson, "A vision-based pipeline for vehicle counting, speed estimation, and classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 12, pp. 7547–7560, Dec. 2021, doi: 10.1109/TITS.2020.3004066.

[16] J. Sochor, R. Juránek, and A. Herout, "Traffic surveillance camera calibration by 3D model bounding box alignment for accurate vehicle speed measurement," *Comput. Vis. Image Understand.*, vol. 161, pp. 87–98, Aug. 2017, doi: 10.1016/J.CVIU.2017.05.015.

[17] D. Bell, W. Xiao, and P. James, "Accurate vehicle speed estimation from monocular camera footage," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 2020, pp. 419–426, Aug. 2020, doi: 10.5194/isprs-annals-V-2-2020-419-2020.

[18] H. Rodríguez-Rangel, L. A. Morales-Rosales, R. Imperial-Rojo, M. A. Roman-Garay, G. E. Peralta-Peñuñuri, and M. Lobato-Báez, "Analysis of statistical and artificial intelligence algorithms for real-time speed estimation based on vehicle detection with YOLO," *Appl. Sci.*, vol. 12, no. 6, p. 2907, Mar. 2022, doi: 10.3390/APP12062907.

[19] A. Cvijetic, S. Djukanovic, and A. Perunicic, "Deep learning-based vehicle speed estimation using the YOLO detector and 1D-CNN," in *Proc. 27th Int. Conf. Inf. Technol. (IT)*, Zabljak, Montenegro, Feb. 2023, pp. 1–4, doi: 10.1109/IT57431.2023.10078518.

[20] A. Perunicic, S. Djukanovic, and A. Cvijetic, "Vision-based vehicle speed estimation using the YOLO detector and RNN," in *Proc. 27th Int. Conf. Inf. Technol. (IT)*, Zabljak, Montenegro, Feb. 2023, pp. 1–4, doi: 10.1109/IT57431.2023.10078639.

[21] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[22] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Beijing, China, Sep. 2017, pp. 3645–3649, doi: 10.1109/ICIP.2017.8296962.

[23] J. Shi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Seattle, WA, USA, Jun. 1994, pp. 593–600, doi: 10.1109/CVPR.1994.323794.

[24] J. Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm," Microprocessor Res. Lab., Intel Corp., Santa Clara, CA, USA, Tech. Rep., 2000. [Online]. Available: http://robots.stanford.edu/cs223b04/algo_affine_tracking.pdf

[25] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Proc. Conf. Eur. Conf. Comput. Vis. (ECCV)*, Berlin, Germany, 2006, pp. 404–417.

[26] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory Appl. (VISSAPP)*, Lisboa, Portugal, Feb. 2009, pp. 331–340.

[27] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[28] S. Javadi, M. Dahl, and M. I. Pettersson, "Vehicle speed measurement model for video-based systems," *Comput. Electr. Eng.*, vol. 76, pp. 238–248, Jun. 2019, doi: 10.1016/j.compeleceng.2019.04.001.

[29] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.

[30] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017, *arXiv:1708.02002*.

[31] J. Xie, Y. Zheng, R. Du, W. Xiong, Y. Cao, Z. Ma, D. Cao, and J. Guo, "Deep learning-based computer vision for surveillance in ITS: Evaluation of state-of-the-art methods," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3027–3042, Apr. 2021, doi: 10.1109/TVT.2021.3065250.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, *arXiv:1506.01497*.

[33] A. Kumar, P. Khorramshahi, W.-A. Lin, P. Dhar, J.-C. Chen, and R. Chellappa, "A semi-automatic 2D solution for vehicle speed estimation from monocular videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Salt Lake City, UT, USA, Jun. 2018, pp. 137–1377, doi: 10.1109/CVPRW.2018.00026.

[34] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, Sep. 2016, pp. 3464–3468, doi: 10.1109/ICIP.2016.7533003.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," 2017, *arXiv:1703.06870*.

[36] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[37] G. Jocher. (2022). *YOLOv5 by Ultralytics*. [Online]. Available: https://github.com/ultralytics/yolov5/releases

[38] K.-J. Kim, P.-K. Kim, Y.-S. Chung, and D.-H. Choi, "Multi-scale detector for accurate vehicle detection in traffic surveillance data," *IEEE Access*, vol. 7, pp. 78311–78319, 2019, doi: 10.1109/ACCESS.2019.2922479.

[39] A. Horzyk and E. Ergun, "YOLOv3 precision improvement by the weighted centers of confidence selection," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Glasgow, U.K., Jul. 2020, pp. 1–8, doi: 10.1109/IJCNN48605.2020.9206848.

[40] M. A. B. Zuraimi and F. H. K. Zaman, "Vehicle detection and tracking using YOLO and DeepSORT," in *Proc. IEEE 11th IEEE Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, Penang, Malaysia, Apr. 2021, pp. 23–29, doi: 10.1109/ISCAIE51753.2021.9431784.
[41] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: Challenges, architectural successors, datasets and applications," *Multimedia Tools Appl.*, vol. 82, no. 6, pp. 9243–9275, Aug. 2022, doi: 10.1007/s11042-022-13644-y.
[42] U. Sirisha, S. P. Praveen, P. N. Srinivasu, P. Barsocchi, and A. K. Bhoi, "Statistical analysis of design aspects of various YOLO-based deep learning models for object detection," *Int. J. Comput. Intell. Syst.*, vol. 16, no. 1, Aug. 2023, Art. no. 126, doi: 10.1007/s44196-023-00302-w.
[43] C.-Y. Wang, A. Bochkovskiy, and H.-Y.-M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Vancouver, BC, Canada, Jun. 2023, pp. 7464–7475, doi: 10.1109/CVPR52729.2023.00721.
[44] H. Dong, M. Wen, and Z. Yang, "Vehicle speed estimation based on 3D ConvNets and non-local blocks," *Future Internet*, vol. 11, no. 6, p. 123, May 2019, doi: 10.3390/fi11060123.

**KEATTISAK SANGSUWAN** received the B.Eng. and M.Eng. degrees in electronics engineering from the King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 2003 and 2005, respectively, the M.B.A. degree from Sukhothai Thammathirat Open University, Thailand, in 2009, and the M.Sc. degree in aeronautical and space systems (embedded systems) from Institut Supérieur de l'Aéronautique et de l'Espace, Toulouse, France, in 2014. He is currently pursuing the Ph.D. degree in microelectronics and embedded systems with the School of Engineering and Technology, Asian Institute of Technology, Thailand. His research interests include image processing, machine learning, and embedded systems.

**MONGKOL EKPANYAPONG** received the B.Eng. degree in computer engineering from Chulalongkorn Univerisity, Bangkok, Thailand, in 1997, the M.Eng. degree in computer science from the Asian Institute of Technology, Thailand, in 2000, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2003 and 2006, respectively. From 1997 to 1998, he was a System Engineer with United Communication Network, Thailand. From 2006 to 2009, he was a Senior Computer Architect with the Core 2 Architecture Design Team, Intel Corporation, USA. He joined the School of Engineering and Technology, Asian Institute of Technology, in 2009, where he is currently an Associate Professor. His research interests include VLSI design, physical design automation, microarchitecture, compilers, and embedded systems.